Constructing MACs using blockciphers that are only secure as MACs

Yevgeniy Dodis (NYU), John P. Steinberger (UBC)

August 18, 2009

Message Authentication Codes (MACs)

- A MAC is a function requiring a secret key to compute

Message Authentication Codes (MACs)

- A MAC is a function requiring a secret key to compute
- Enables proof of knowledge of key (signatures, etc)

Message Authentication Codes (MACs)

- A MAC is a function requiring a secret key to compute
- Enables proof of knowledge of key (signatures, etc)
- Must be resistant to chosen message attack

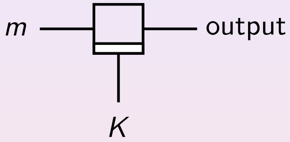
$$f_K(x_1) \quad f_K(x_2) \quad \cdots \quad f_K(x_q)$$

$$A$$

$$\longrightarrow (x_{q+1}, y) \text{ s.t. } f_K(x_{q+1}) = y$$

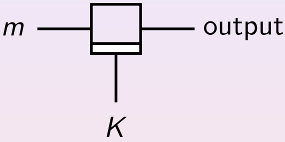
Blockcipher as a MAC

- Blockcipher key = secret key



Blockcipher as a MAC

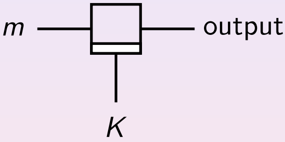
- Blockcipher key = secret key



- Is a fixed input length MAC

Blockcipher as a MAC

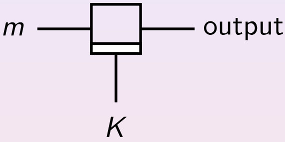
- Blockcipher key = secret key



- Is a fixed input length MAC
- Blockciphers typically modeled as PRP's, much stronger than being a MAC

Blockcipher as a MAC

- Blockcipher key = secret key



- Is a fixed input length MAC
- Blockciphers typically modeled as PRP's, much stronger than being a MAC
- MACs only need to be unpredictable and not pseudorandom

Goal and Result

- Construct a variable input length blockcipher-based MAC whose security can be proved from the MAC security of the underlying blockcipher.

Goal and Result

- Construct a variable input length blockcipher-based MAC whose security can be proved from the MAC security of the underlying blockcipher.
- The secret key(s) of the MAC are the blockcipher keys (in particular, the blockciphers are used in fixed key mode)

Goal and Result

- Construct a variable input length blockcipher-based MAC whose security can be proved from the MAC security of the underlying blockcipher.
- The secret key(s) of the MAC are the blockcipher keys (in particular, the blockciphers are used in fixed key mode)
- Impediment: The blockcipher can have MAC-insecurity as low as $1/2^n$, but an iterated, non-wide-pipe, arbitrary domain scheme can only have MAC insecurity as low as $q^2/2^n$ ("birthday barrier")

Goal and Result

- Construct a variable input length blockcipher-based MAC whose security can be proved from the MAC security of the underlying blockcipher.
- The secret key(s) of the MAC are the blockcipher keys (in particular, the blockciphers are used in fixed key mode)
- Impediment: The blockcipher can have MAC-insecurity as low as $1/2^n$, but an iterated, non-wide-pipe, arbitrary domain scheme can only have MAC insecurity as low as $q^2/2^n$ ("birthday barrier")

Result: A rate 3 variable input length MAC function whose security is at most $q^2 \log^2(q)$ worse than the MAC security of the underlying blockcipher.

Previous Constructions

- Many constructions secure under pseudorandomness of f fail (in general) assuming only unpredictability.
  - Includes CBC-MAC [AB99], HMAC, hash-then-mac using universal hash functions, contant-round Feistel Network [AB99,DP07],...

Previous Constructions

- Many constructions secure under pseudorandomness of f fail (in general) assuming only unpredictability.
  - Includes CBC-MAC [AB99], HMAC, hash-then-mac using universal hash functions, contant-round Feistel Network [AB99,DP07],...
- Some are theoretically secure, but inefficient:
  - Includes generic MAC-to-PRF solutions [GL89,NR98], many-round Feistel [DP07], hash-then-sign using collision-resistant hash functions,...

Previous Constructions

- Many constructions secure under pseudorandomness of f fail (in general) assuming only unpredictability.
  - Includes CBC-MAC [AB99], HMAC, hash-then-mac using universal hash functions, contant-round Feistel Network [AB99,DP07],...
- Some are theoretically secure, but inefficient:
  - Includes generic MAC-to-PRF solutions [GL89,NR98], many-round Feistel [DP07], hash-then-sign using collision-resistant hash functions,...
- Previous Best: The rate 2 "enciphered CBC MAC" of Dodis, Pietrzak and Puniya (Eurocrypt 08) whose security is q^4 worse than the MAC security of the underlying blockcipher (q is number of queries).

Additional Features

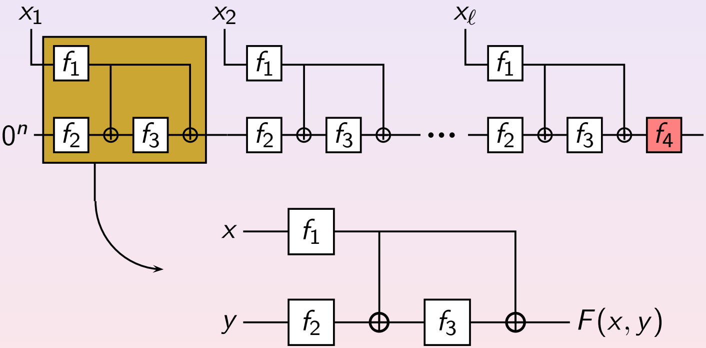
(1) PRF preservation: If the blockcipher is a PRP the mode is a PRF (with birthday security)

Additional Features

(1) PRF preservation: If the blockcipher is a PRP the mode is a PRF (with birthday security)

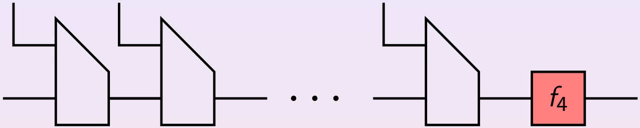
(2) Still indistinguishable from PRF even when the adversary is allowed to make "transcript queries" showing all blockcipher query data $\Longrightarrow$ the hash function can have a completely leaky implementation as long as the blockcipher keys aren't leaked

Our Construction



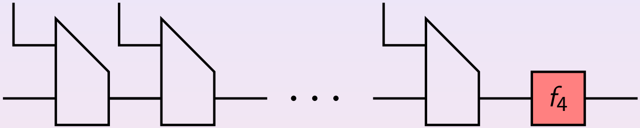
Shrimpton-Stam '07

Proof Framework (An and Bellare 99)



- To forge, the adversary must either find an internal collision or forge f_4.

Proof Framework (An and Bellare 99)



- To forge, the adversary must either find an internal collision or forge f_4.
- Main task is therefore to bound the collision resistance of the compression function using only the MAC security of the underlying blockcipher

Main Theorem

The collision resistance of the Shrimpton-Stam compression function is at most $O(q^2 \log^2(q))$ worse than the MAC security of the blockcipher.

Main Theorem

The collision resistance of the Shrimpton-Stam compression function is at most $O(q^2 \log^2(q))$ worse than the MAC security of the blockcipher.

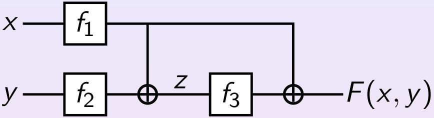
- Proof Strategy: Given a collision-finding adversary A that has advantage ε, exhibit a MAC-forging adversary B for the blockcipher with advantage $\varepsilon / q^2 \log^2(q)$.

Main Theorem

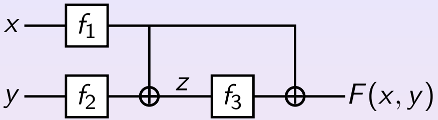
The collision resistance of the Shrimpton-Stam compression function is at most $O(q^2 \log^2(q))$ worse than the MAC security of the blockcipher.

- Proof Strategy: Given a collision-finding adversary A that has advantage ε, exhibit a MAC-forging adversary B for the blockcipher with advantage $\varepsilon / q^2 \log^2(q)$.
- Comparison to SS'09: information-theoretic argument assuming perfectly random f_i's versus a computational reduction from one type of adversary to another.

Proof of Main Theorem

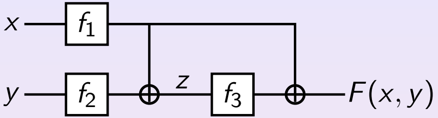


Proof of Main Theorem



- Simplifying Assumption: Queries to f_1, f_2 come before queries to f_3.

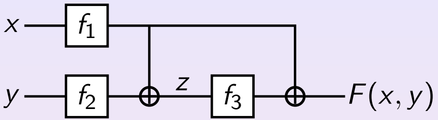
Proof of Main Theorem



- Simplifying Assumption: Queries to f_1, f_2 come before queries to f_3.

Definition: For each $z \in \{0,1\}^n$ let $\mathrm{Pairs}(z) = \{(x, y)$ s.t. $f_1(x) \oplus f_2(y) = z$, A has made the queries $f_1(x), f_2(y)\}$.

Proof of Main Theorem



- Simplifying Assumption: Queries to f_1, f_2 come before queries to f_3.

Definition: For each $z \in \{0,1\}^n$ let $\text{Pairs}(z) = \{(x, y)$ s.t. $f_1(x) \oplus f_2(y) = z, A$ has made the queries $f_1(x), f_2(y)\}$.

Observation: If f_1, f_2 are behaving randomly then (i) $C := \max_z |\text{Pairs}(z)|$ is small, (ii) with each query $f_3(z)$, A learns at most $|\text{Pairs}(z)| \leq C$ new values $F(x, y)$, (iii) A learns at most Cq values $F(x, y)$ total.

Proof of Main Theorem

Strategy 1: If f_1, f_2 are behaving randomly then B can guess the answer to a query $f_3(z)$ by guessing that $F(x, y) = F(x', y')$ for some $(x, y) \in \mathrm{Pairs}(z)$ and some (x', y') for which $F(x', y')$ is already known. More precisely, since $F(x, y) = f_1(x) \oplus f_3(z)$, guess $f_3(z) = f_1(x) \oplus f_1(x') \oplus f_3(z')$.

Proof of Main Theorem

Strategy 1: If f_1, f_2 are behaving randomly then B can guess the answer to a query $f_3(z)$ by guessing that $F(x,y) = F(x',y')$ for some $(x,y) \in \mathrm{Pairs}(z)$ and some (x',y') for which $F(x',y')$ is already known. More precisely, since $F(x,y) = f_1(x) \oplus f_3(z)$, guess $f_3(z) = f_1(x) \oplus f_1(x') \oplus f_3(z')$.

- Here B has probability of success $\varepsilon \frac{1}{q} \frac{1}{C} \frac{1}{Cq} = \varepsilon / q^2 C^2$, acceptable as long as $C \leq \log(q)$.

Proof of Main Theorem

Strategy 1: If f_1, f_2 are behaving randomly then B can guess the answer to a query $f_3(z)$ by guessing that $F(x, y) = F(x', y')$ for some $(x, y) \in \text{Pairs}(z)$ and some (x', y') for which $F(x', y')$ is already known. More precisely, since $F(x, y) = f_1(x) \oplus f_3(z)$, guess $f_3(z) = f_1(x) \oplus f_1(x') \oplus f_3(z')$.

- Here B has probability of success $\varepsilon \frac{1}{q} \frac{1}{C} \frac{1}{Cq} = \varepsilon / q^2 C^2$, acceptable as long as $C \leq \log(q)$.

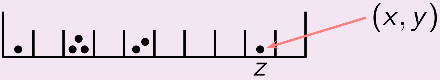
Strategy 2: If f_1, f_2 are not behaving randomly and $|\text{Pairs}(z)| > \log(q)$ for some z, then use the non-randomness of f_1, f_2 to forge either f_1 or f_2.

Exploiting the non-randomness of f_1, f_2

- Will display a strategy for B that forges f_1 or f_2 with probability $1/4q^2$ whenever $|\mathrm{Pairs}(z)| > \log(q)$ for some z.

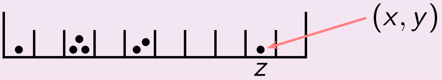
Exploiting the non-randomness of f_1, f_2

- Will display a strategy for B that forges f_1 or f_2 with probability $1/4q^2$ whenever $|\mathrm{Pairs}(z)| > \log(q)$ for some z.
- View each value of $z \in \{0,1\}^n$ as a bin and points $(x, y) \in \mathrm{Pairs}(z)$ as balls placed in these bins.



Exploiting the non-randomness of f_1, f_2

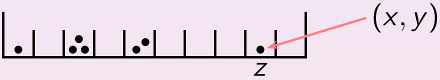
- Will display a strategy for B that forges f_1 or f_2 with probability $1/4q^2$ whenever $|\mathrm{Pairs}(z)| > \log(q)$ for some z.
- View each value of $z \in \{0,1\}^n$ as a bin and points $(x, y) \in \mathrm{Pairs}(z)$ as balls placed in these bins.



- With each query made by A, as many as q new balls are placed into the bins. In total, q^2 balls are placed.

Exploiting the non-randomness of f_1, f_2

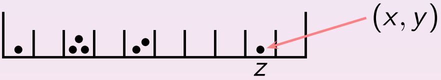
- Will display a strategy for B that forges f_1 or f_2 with probability $1/4q^2$ whenever $|\mathrm{Pairs}(z)| > \log(q)$ for some z.
- View each value of $z \in \{0,1\}^n$ as a bin and points $(x, y) \in \mathrm{Pairs}(z)$ as balls placed in these bins.



- With each query made by A, as many as q new balls are placed into the bins. In total, q^2 balls are placed.
- B's task is to predict the bin of some ball.

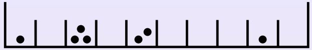
Exploiting the non-randomness of f_1, f_2

- Will display a strategy for B that forges f_1 or f_2 with probability $1/4q^2$ whenever $|\mathrm{Pairs}(z)| > \log(q)$ for some z.
- View each value of $z \in \{0,1\}^n$ as a bin and points $(x, y) \in \mathrm{Pairs}(z)$ as balls placed in these bins.



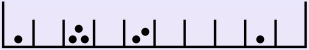
- With each query made by A, as many as q new balls are placed into the bins. In total, q^2 balls are placed.
- B's task is to predict the bin of some ball.
- Can reduce to the case where the balls are thrown one by one.

Balls-in-Bins Game



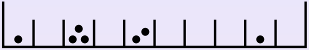
- q^2 balls are placed by A into 2^n bins such that some bin receives $> \log(q)$ balls; B must predict the position of a ball with probability at least $1/4q^2$.

Balls-in-Bins Game



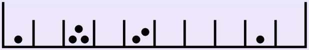
- q^2 balls are placed by A into 2^n bins such that some bin receives $> \log(q)$ balls; B must predict the position of a ball with probability at least $1/4q^2$.
- B chooses an index $i \in \{1, \ldots, q^2\}$ and a "weight" $t \in \{1, \ldots, \log(q)\}$, at random. When the i-th ball is thrown, B guesses a bin that has at least t balls.

Balls-in-Bins Game



- q^2 balls are placed by A into 2^n bins such that some bin receives $> \log(q)$ balls; B must predict the position of a ball with probability at least $1/4q^2$.
- B chooses an index $i \in \{1, \ldots, q^2\}$ and a "weight" $t \in \{1, \ldots, \log(q)\}$, at random. When the i-th ball is thrown, B guesses a bin that has at least t balls.
- To win, B must (i) make its guess for a ball that is thrown into a bin with $\geq t$ balls, and (ii) choose the right bin among these.

Balls-in-Bins Game



- q^2 balls are placed by A into 2^n bins such that some bin receives $> \log(q)$ balls; B must predict the position of a ball with probability at least $1/4q^2$.
- B chooses an index $i \in \{1, \ldots, q^2\}$ and a "weight" $t \in \{1, \ldots, \log(q)\}$, at random. When the i-th ball is thrown, B guesses a bin that has at least t balls.
- To win, B must (i) make its guess for a ball that is thrown into a bin with $\geq t$ balls, and (ii) choose the right bin among these.
- Let c_j = total number of balls thrown into bins with $\geq j$ balls in them already. Then for fixed t, B chance's of winning is $\geq \frac{c_t}{q^2} \frac{1}{c_{t-1}} = \frac{1}{q^2} \frac{c_t}{c_{t-1}}$.

Cute Computation

B's chance of winning is

$$
\begin{aligned}
\sum_{t=1}^{\log(q)} \frac{1}{\log(q)} \frac{1}{q^2} \frac{c_t}{c_{t-1}} &= \frac{1}{q^2} \text{ ArithmeticMean} \left(\frac{c_1}{c_0}, \ldots, \frac{c_{\log(q)}}{c_{\log(q)-1}} \right) \\
&\geq \frac{1}{q^2} \text{ GeometricMean} \left(\frac{c_1}{c_0}, \ldots, \frac{c_{\log(q)}}{c_{\log(q)-1}} \right) \\
&= \frac{1}{q^2} \left(\frac{c_{\log(q)}}{c_0} \right)^{\frac{1}{\log(q)}} \geq \frac{1}{q^2} \left(\frac{1}{q^2} \right)^{\frac{1}{\log(q)}} \\
&= \frac{1}{q^2} \frac{1}{2^{\log(q^2)\frac{1}{\log(q)}}} \\
&= \frac{1}{4q^2}
\end{aligned}
$$

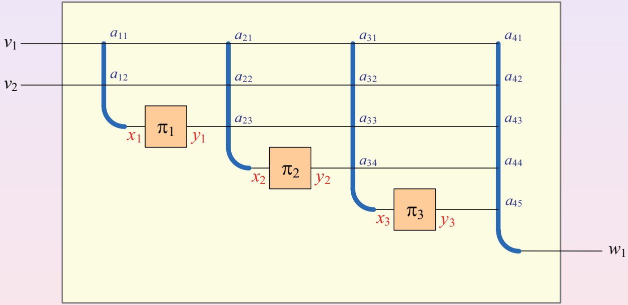
QED.

Theorem (Take-Away Fact)

If Q objects are sequentially placed into infinitely many slots such that some slot accumulates more than $\log(Q)$ objects by the end of the process, it is possible to forecast the position of one of the objects with probability at least $1/Q$.

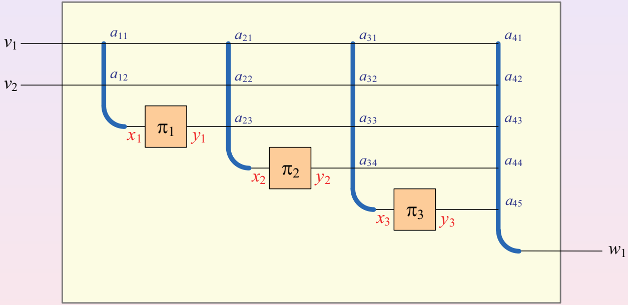
Open Questions/Remarks

- Can also use the LP231 compression function (Rogaway/S08):



Open Questions/Remarks

- Can also use the LP231 compression function (Rogaway/S08):



- Open question: going beyond birthday security